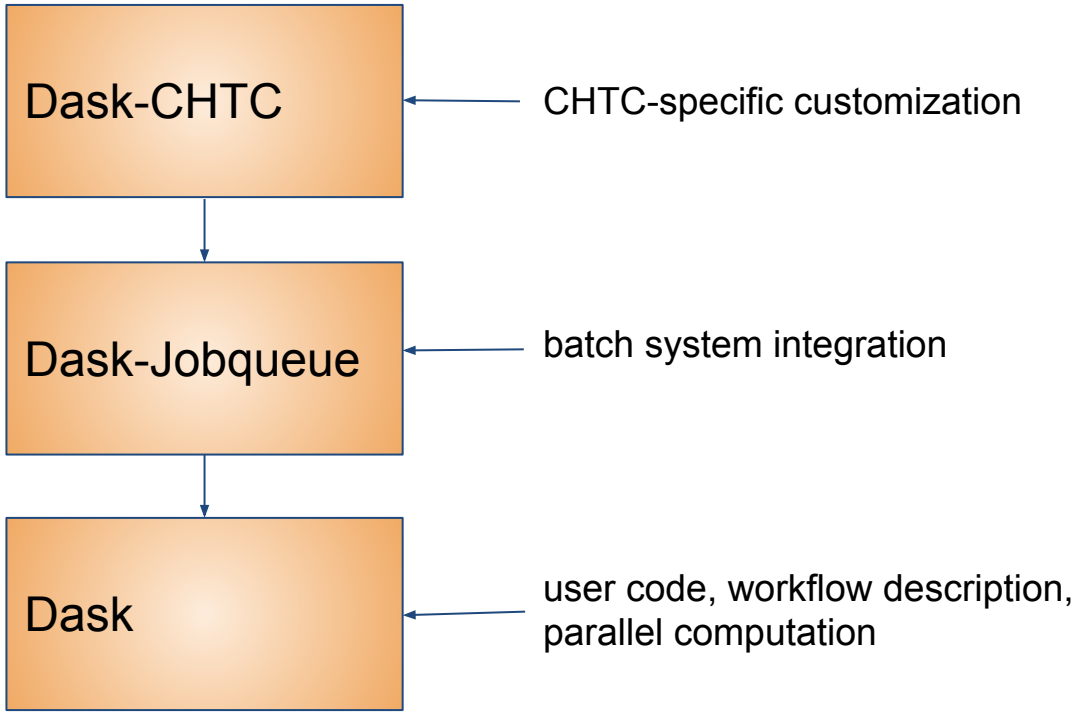
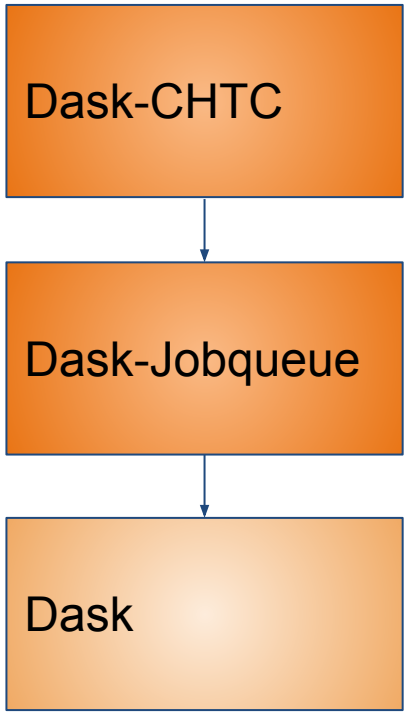
Lightweight Site-Specific Dask Integration for HTCondor at CHTC

Mátyás Selmeci, Josh Karpel, Brian Bockelman

Integrating Dask with HTCondor in CHTC involves multiple layers of software

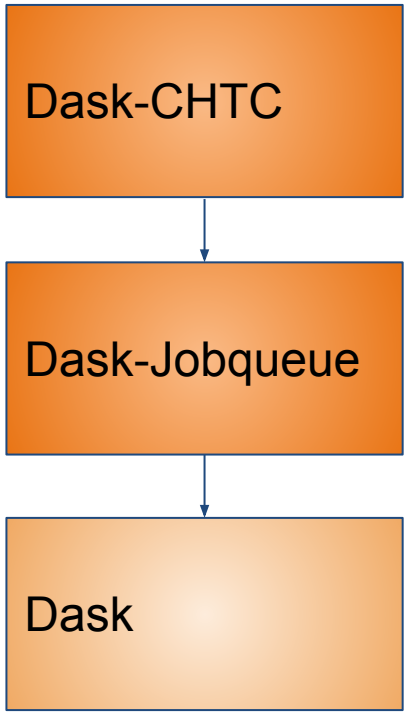


Dask is a Python library for building up and executing parallel workflows



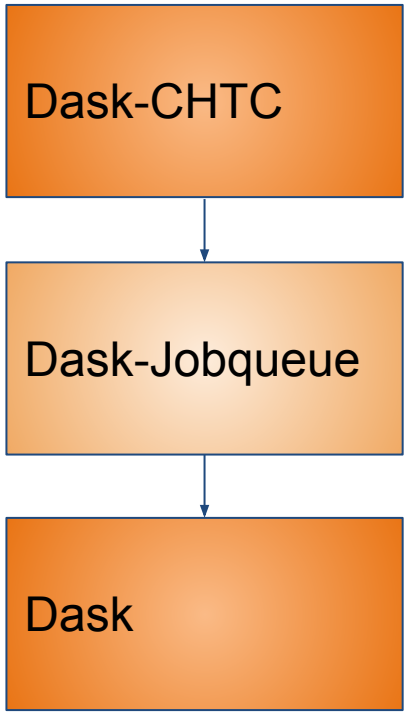
- Dask defines data structures compatible with NumPy arrays, Pandas dataframes, etc.
- When using these, Dask will:
  - Split the data structures into chunks (e.g. an array into multiple smaller arrays)
  - Turn your operations (e.g. array multiplication) into smaller tasks that operate on these chunks
  - Build up a workflow of tasks with dependencies (a DAG)
  - Send these tasks to separate processes to be executed
- Addons for ML frameworks like PyTorch

Dask runs tasks in parallel with a scheduler/worker system



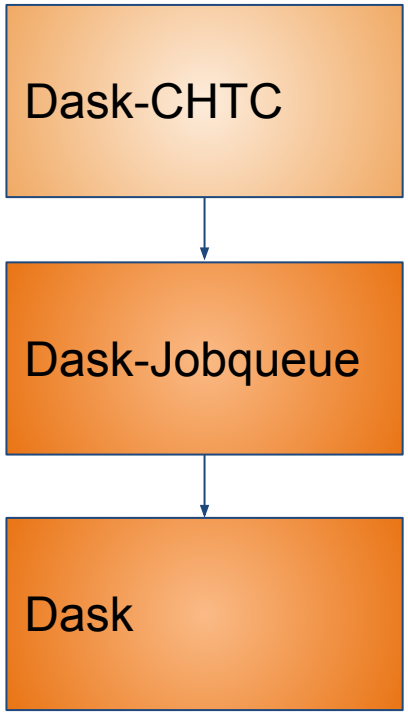
- A central *scheduler* lives in the same process as the user's code and distributes tasks
- *Workers* live in separate processes and actually perform the tasks
- Processes communicate over TCP; this means they can run on separate machines
- Workers can come and go; schedulers will rebalance tasks

Dask-Jobqueue runs workers as jobs on batch systems



- Originally for HPC batch systems (e.g. PBS, SLURM) with a lot of environment assumptions e.g. shared filesystem so necessary libraries are on the execute node
- HTCondor support was added in 2019, but with same assumptions (for simplicity)
- But… CHTC doesn't want to use a shared file system

Dask-CHTC customizes Dask-Jobqueue to fit CHTC's needs



- Transfers libraries and executables to avoid shared filesystem
- Adds to Requirements so workers only run on specific machines
- Adds custom parameters, e.g. "gpus=1" to request a GPU for the worker
- Adds classad attributes so the jobs can be tracked ("IsDaskWorker=True")

Dask-CHTC uses Docker to work around need for a shared file system

- Put the environment into a Docker container, and use HTCondor's Docker Universe to run the dask-worker in the container
- Use HTCondor file transfer to send files that are not in the image
- Dask developers provide a basic image for running a worker, or users can customize or write their own (we provide docs)

Live demo

(adapted from the Dask Arrays example)

Future work

- Dask Scheduler can be resource intensive - move it off of the submit node

- Extract and share features that have a wider audience

Related work

Coffea-Casa

- Prototype analysis facility for USCMS developed by a team at University of Nebraska
- Solved many of the same problems and provided a basis for our work
- PyHEP 2020 video
- PyHEP 2020 materials

Links

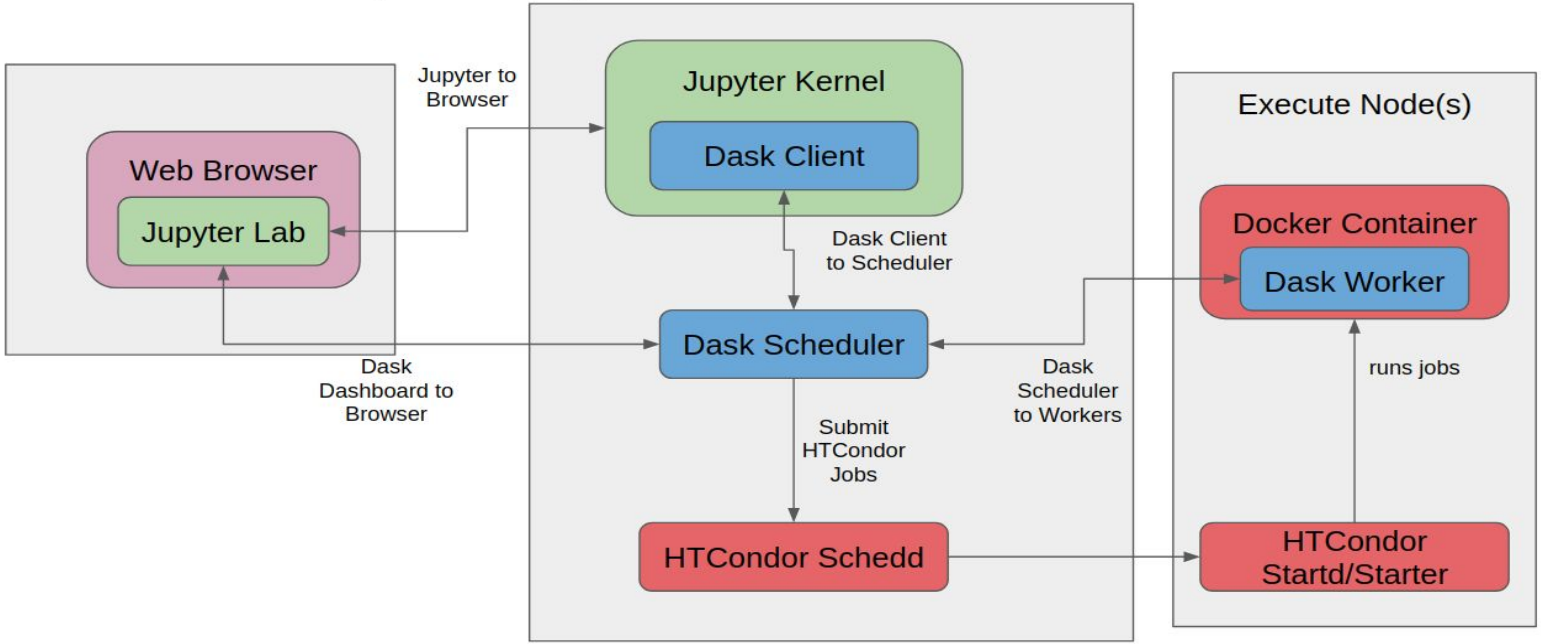
- Dask home page
- Dask-jobqueue home page
- Dask-CHTC documentation
- Dask-CHTC GitHub repo

Backup slides

Docker adds a challenge to networking

- Containers are isolated from their host's network:
  - They have their own IP addresses that external hosts don't know
  - They do not know the IP address of their own host
- Extract that information from the job classad:
  - $_CONDOR_JOB_AD is a path to a file containing the job classad
  - RemoteHost contains the name of the host the container is executing on
  - dask_HostPort contains the *outside* port from which the worker is accessible
- … and tell the worker to advertise this public address to the scheduler

Kernel, Client, Scheduler on Submit Node



Credit: Josh Karpel